10

Resumen

Transcripción

[00:00] Sé que ha sido mucha información en esta clase sobre todo porque hemos intentado pues romper ese concepto un poco de lo que es herencia en la utilización de código e introducir lo que es el nuevo concepto de interfaces que yo sé que no es nada fácil de entender al inicio.

[00:23] Entonces vamos a dar un pequeño overview de cuál era el problema que teníamos. El primer problema era que estas tres relaciones no existían. Perfecto. Entonces, lo que teníamos era un acoplamiento a nivel de clases de autenticable con funcionario. Todo objeto autenticable era a su vez un funcionario.

[00:54] Incluso voy a hacer una pequeña flecha aquí para que se vea la relación. Vamos a cambiarle el color, listo. Ese es el problema de antes. Todo objeto autenticable era un funcionario. Y si era autenticable, yo podía accesar al sistema interno. Incluso para mejorar esto un poco más voy a borrar esto aquí, y esto de aquí se va.

[01:26] El sistema interno va a entrar aquí como un mediador por así decirlo, un objeto intermedio, independiente pero intermedio. Entonces, a través de él, todos los objetos que son autenticables van a iniciar sesión. ¿Pero cuál es el problema? Que todo autenticable era un funcionario y a nivel de clases tú solo puedes hacer extends de una sola clase, sea abstracta o sea normal.

[01:58] Entonces, ¿la interfaz dónde es que entra? Entra para desacoplar todo lo que es autenticable de lo que es funcionario. ¿Cómo así? Porque ya que es el

sistema interno, es totalmente independiente, esto de aquí es un dominio totalmente diferente de lo que es el dominio del funcionario en sí, porque vemos que aquí ya entra un actor que es el cliente, y el cliente no tiene nada que ver con los funcionarios ni con los gerentes ni administradores.

[02:37] Entonces conceptualmente tú no puedes darle características a un objeto que no le pertenecen solamente para ahorrar código, este es la primera característica. Vamos a pasar esto un poco más al centro. ¿Por qué? Porque si bien de este lado extendemos, aquí extienden, extienden, extienden. También implementamos, que es la otra palabra reservada que sería aquí, vamos a dar un espacio para lo que sería implements que es la otra palabra reservada de Java.

[03:15] Para explicar que él está implementando esta interfaz. Ahora, viéndolo ya a nivel del código, vamos a analizar bien la interfaz, solo para hacer una overview de lo que ya vimos. Una interfaz es muy similar a una clase abstracta, solamente que en la interfaz todos los métodos son abstractos, de clase abstracta, puede tener métodos abstractos y métodos implementados. Ese es un plus de la clase abstracta.

[03:45] La clase abstracta puede tener variables, la interfaz no. Vamos a hacerlo además y vamos a nuestra clase funcionario para verlo mejor. Clase abstracta puede tener atributos privados, interfaz no puede. Clase abstracta puede tener constructor, la interfaz no puede tener constructor. ¿Por qué? Porque no es un objeto en sí, es un tipo, es un rótulo, es una etiqueta que le das al objeto pero no se una representación del objeto en sí.

[04:19] ¿La interfaz puede tener métodos implementados? No. La interfaz solamente tiene la firma del método y ya la implementación es problema de cada objeto que implementa esa interfaz. Entonces, ¿qué nos permite la interfaz?

[04:38] Dado que la interfaz la estamos viendo como si fuera un sello, un rótulo que le damos a ciertos objetos o no, la interfaz no representa digamos un nivel jerárquico a nivel de diagrama, por ejemplo autenticable. Representa sí una característica de estos tres objetos, pero conceptualmente hablando no representa una relación de padre e hijo porque tú no dices por ejemplo que administrador es hijo de autenticable. ¿Por qué?

[05:12] Porque no extiende de autenticable. Administrador es hijo de funcionario porque administrador es un funcionario. A nivel conceptual, porque funcionario tiene características como nombre, salario y lo demás, que también son usadas por el administrador, pero en el caso de cliente, cliente solamente es autenticable, y autenticable es una característica más.

[05:44] Entonces sí me gustaría que practiquen lo que es digamos la herencia en sí, conceptualmente hablando. Si bien la herencia nos ayuda a reutilizar el código no es bueno utilizarlo solo para eso. Recuerden que tienen que tener un significado, la herencia significa una cosa, la herencia es una relación, un relacionamiento familiar justamente. Administrador, gerente y contador son de la misma familia.

[06:13] Cliente no es de esa familia. Cliente solamente es autenticable. Al igual que de repente cliente, administrador y gerente los tres son autenticables y contador no lo es. Entonces ejerciten bastante esa abstracción, identificar grupos de objetos con relaciones entre sí, relaciones de parentesco, entonces es un buen punto, es un buen caso para aplicar herencia.

[06:40] En el caso de que no sea parentesco sino una característica, por ejemplo en tu grupo de amigos, por ejemplo, cada uno tiene su propia familia, ¿cierto? Cada uno tiene su propia familia en casa, pero en tu grupo de amigos se juntan y todos comparten algo en común. No son una familia, pero comparten algo en común.

[07:07] Entonces ese algo en común es la interfaz. Digamos, algo en común que tienen es que tu grupo de amigo solo usan polos verdes. Es así, cada vez que salen, todos se ponen polos verdes. Entonces cada uno tiene su familia, cada uno es hijo de Pepito, fulanito, etcétera. Pero la interfaz sería "usas polo verde" o polo verde mejor dicho y el método sería "usas polo verde". ¿Por qué?

[07:30] Porque si bien no eres familia con tus amigos, tienes algo en común, tienes algo ahí, un interés en común. ¿Cuál es ese interés? Tu polo verde. Es exactamente lo mismo con la interfaz. No representa una relación de familiaridad o de relación directa pero es una característica que necesitas compartir y esa característica se puede enviar como parámetro a los métodos. ¿Por qué?

[07:58] Si vamos nuevamente al ejemplo que se me ocurre ahorita del polo verde, voy a entrar aquí a sistema interno y es como decir: "aquí se autentican todos los que tengan polo verde", así de simple. Entonces Java entiende que autenticable, al ser una característica compartida por uno o más objetos, porque es una interfaz, entonces también tiene sus propios atributos.

[08:26] Las dudas, recuerden, el foro está abierto 24/7, yo intento responder lo más rápido que puedo a cualquier pregunta que tengan, y como siempre les digo: "practiquen, practiquen, practiquen". Yo sé que esto es un poco más complicado que lo que hemos visto anteriormente pero ya van a ver la gran utilizad que tiene esto en el próximo video. Nos vemos. Chau, chau.